**UDH Web框架简介**

**----Ember.js**

目录

[1.相关概念 1](#_Toc417979262)

[2.模块详解 2](#_Toc417979263)

[2.1.应用 2](#_Toc417979264)

[2.2.index.html 2](#_Toc417979265)

[2.3.模板 3](#_Toc417979266)

[2.4.路由器 3](#_Toc417979267)

[2.5.控制器 3](#_Toc417979268)

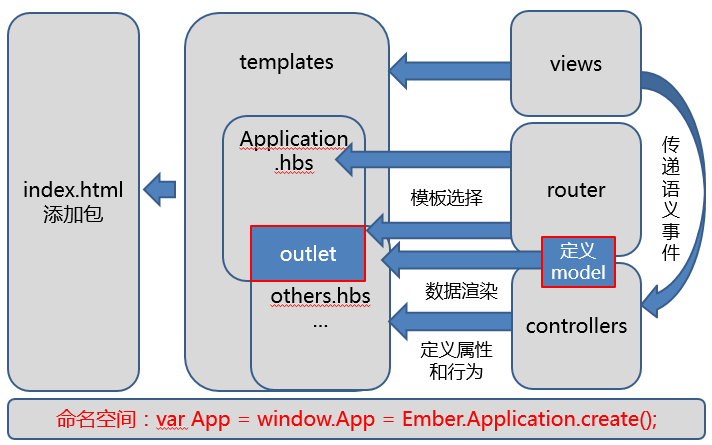
[2.6.模型 3](#_Toc417979269)

[2.7.视图 4](#_Toc417979270)

UDH的Web框架基于Ember.js搭建。Ember.js是基于javascript的一个开源的MVC web框架，它能够将本地GUI框架中的工具和概念与Web应用的URL结合在一起，致力于建立强大而丰富的web界面。

# 相关概念

MVC的web框架一般具有模型、视图以及控制器三部分，ember也不例外，但是，相比于其他的MVC框架，它具有更丰富的模块化机制，一般包括六种模块：模板、路由、控制器、视图、模型以及应用，另外，一个完整的ember web应用还需要一个唯一的index.html。



其中各个模块的关系由上图所示。

# 模块详解

## 应用

**应用**，即命名空间， 整个ember框架中仅包含一个应用，相当于生成一个ember的应用实例：



只有生成了ember的应用实例，该实例的模块才是有意义的，如路由：App.ApplicationRoute，控制器:App.ApplicationController，视图：App.ListView都需要依据此命名。

## index.html

整个web应用只包含一个html文件，通常命名为index.html。因此，可以说ember是一个单页面的web应用。 index.html文件中通常并不包含实际的代码，仅引入一些必须的javascript文件。在index.html之上，通过添加不同的模板来显示界面。

## 模板

**模板**主要用于渲染显示给用户的界面， 主要基于Handlebars模板语言编写，与纯HTML较为类似，只是定义了一些其特有的**表达**：

例如表达式{{serviceName}},该语句可以实现从一个模型中获取一个服务名并将该服务名添加到HTML中，并能够实现动态绑定。

每个模板都对应着一个具体的模型，当模型发生改变时，模板也会自动进行更新。

另外，模板还需要一个**出口（Outlets）**, 它类似其他模板的占位符，可以在具有{{outlet}}表达式的模板上添加其他模板，实现模板的叠加，为用户展示更复杂的界面。

应用模板(即图中的Application.hbs)是应用启动的时候默认渲染的模板。应用模版中至少需要一个{{outlet}}占位符，以便路由能根据当前的URL将适当的模版渲染进来。

## 路由器

**路由器**相当于页面的导航，它能够引导用户到达下一个模板显示的内容，例如在UDH中，从“仪表板”页面跳转到“服务”页面，用户能够正确的看到“服务”界面的内容而不是“主机”界面的内容，是路由器在发生作用，它将URL转换为一系列内嵌的有“服务”相关模型数据支撑的模板。因此，当显示给用户的模板和模型发生改变时，Ember自动更新浏览器地址栏中的URL。

由于路由器的作用，用户可以在任意模板上分享链接。当某个用户点击了这个链接时，将看到与分享链接的用户看到的相同内容。

## 控制器

**控制器**，主要用于定义属性和行为，处理由视图或模板直接传递的信息，同时，控制器可以代理一个模型，

控制器可以完成以下工作：

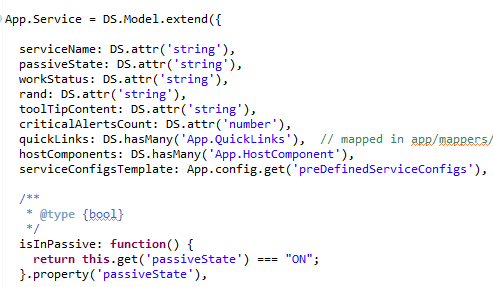
* 为模板代理一个模型，有时可以将模型中要定义的属性放在控制器定义；
* 存放不需要服务器保存的应用属性

Ember.js将这些状态封装在路由器里，这意味着控制器可以更轻量，且更专注一件事。

## 模型

**模型**用于存储**持久化状态数据，**类似于Java中的对象，通常用于构造包含多种属性和行为的数据结构。由上图可以看出，模型既可以在控制器中定义，也可以在路由器中定义。

例如，UDH中定义service模型：



其中主要定义了service的名字、状态、快速链接等的属性，主要用于存储服务相关的数据。

模型通常用于构造应用操作的数据，也用来返回给用户数据。这些对象通常从服务器端加载，并当其在客户端发生改变后又保存到服务器端。

## 视图

**视图，**顾名思义，就是用于为web应用创建可视化，它位于模板之上，可以将一个浏览器中的事件，转换成应用中定义的事件。

通常，Ember.js的视图的创建主要基于以下两个目的：

* 需要处理复杂的用户事件：例如点击事件，可将具体的点击动作传给控制器，让控制器进一步处理，产生相应的操作；
* 构建可重用的组件：例如UDH中各显示指标图表，都是基于同一个父视图进行创建的。

以上是对ember.js的各模块的简单介绍，了解各模块功能有助于创建自己的ember web应用。